From Isomorphism-Based Security for Graphs to Semantics-Preserving Security for the Resource Description Framework (RDF)

by

Zhiyuan Lin

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Resource Description Framework (RDF) is a graph-like data model designed for the web. One of its compelling features is a precise, model-theoretic semantics. We address security in the context of the RDF. We first observe that the problem of securing RDF is related closely to the more traditional problem of securing graphs. Consequently, before we address security for RDF, we make broader contributions to security in the context of graphs. Specifically, we reconcile four different notions of security that have been proposed in prior work, and compare them from the standpoint of strength of security — whether satisfaction of one implies satisfaction of another. We then ask whether strength of security is correlated to computational complexity. We make the somewhat surprising observation that the answer to this question is, "no." We then extend the two strongest notions of security in a natural way for RDF. We establish results on RDF's semantics that then gives us a way of meaningfully quantifying the loss of information from security. Thus, for RDF, we are able to pose the natural trade-off between information-quality and security in a precise way. We show that a corresponding decision problem is **NP**-complete, and with a reduction to CNF-SAT that we have designed and implemented, present empirical results on realistic data. Our empirical results show interesting relationships between the various parameters, such as the size of the graph versus the information-loss, to achieve a particular level of security.

Acknowledgements

I would like to thank my supervisor Professor Tripunitara for his support and guidance.

Dedication

This is dedicated to my parents who have always been there for me.

Table of Contents

List of Figures

Chapter 1

Introduction

In this work, we relate two heretofore distinct topics of research in information security: isomorphism-based anonymization of graphs [24, 27, 28], and semantics-preserving security for the Resource Description Framework (RDF) [9]. In doing so, we make contributions to both. Our intent is to leverage the former for the latter. We discuss our contributions below, under 'Our contributions.'

Graphs and a need for security Graphs are powerful abstractions for representing information. They are used in various contexts, at all levels of hardware and software, from modeling digital Integrated Circuits (ICs), to networks and state machines, and social relationships. Not only are graphs useful as an abstraction, some data can be more naturally stored, and efficiently queried for and retrieved as a graph, rather than, for example, relational database tables. Indeed, there exist database implementations such as Apache Jena [13] that support such graph-based data storage and retrieval.

Notwithstanding how data is modeled and stored, access to it can conflict with the security and privacy of individuals and organizations to whom the data pertains. This is documented extensively in the research literature in security. In the context of graphs, an example is the work of Narayanan and Shmatikov [17], which observes that social-network data that is modeled as a graph can be de-anonymized from the 'structure' of the graph only. Their work has led to proposals of security notions that protect graphs against structural attacks — for example, the work of Zhou and Pei[27] and Wu et al. [24].

Another example, in the rather different context of securing digital ICs, is the work of Imeson et al. [10]. In that work, the threat model is that of a manufacturer that injects malicious logic into the IC. Perceiving the IC as a graph is a first step in devising a manner

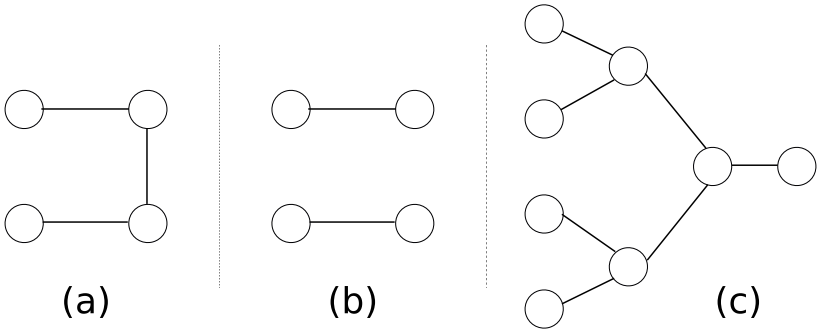

Figure 1.1: Three graphs that help illustrate the four Isomorphism-based notions of security for graphs that we consider. Graphs (a) and (b) are both automorphic. In addition, apart from the identity mapping, there is a mapping between vertices in both graphs such that no vertex is mapped to itself. Graph (a) is subgraph-isomorphic to Graph (b).

in which an IC can be obfuscated before being fabricated to mitigate the threat.

Isomorphism-based notions of Graph-security Common to these pieces of work that propose notions of security in the context of graphs, is that they are based on variants of *graph isomorphism* (Iso). Two graphs $G = \langle V_g, E_g \rangle$ and $H = \langle V_h, E_h \rangle$ are said to be *isomorphic* when there exists an invertible mapping $m\colon V_g \to V_h$ such that $\langle u, v \rangle \in E_g$ if and only if $\langle m(u), m(v) \rangle \in E_h$. (Our notation is customary: V_g and V_h are the sets of vertices of G and H respectively, and E_g and E_h are their sets of edges.)

A graph H is *subgraph isomorphic* to another graph G when a subgraph of H is isomorphic to G. And a graph is said to be (non-trivially) *automorphic* if it is isomorphic to itself via a mapping that is not the identity. In this paper, we focus on the notions of security based on graph isomorphism that are proposed in the work of Zhou and Pei [27], Wu et al. [24], Zou et al. [28] and Imeson et al. [10]. The first three propose notions that are based on isomorphism and automorphism, and the last proposes a notion that is based on subgraph isomorphism. We discuss them in Chapter 3; in Figure 1.1, we illustrate these notions via examples.

As we discuss under 'Graphs and a need for security' above, a motivation for considering such isomorphism-based notions of security for graphs is the observation that when sensitive information is represented as a graph, not only any labels on vertices and edges, but also the structure of the graph can be revealing. This is the case because a reason for representing information as a graph is exactly that a considerable amount of information can be stored in its structure. For example, when vertices correspond to the humans Alice and Bob, it is natural to represent a relationship between them as a edge between those vertices. Thus, even if the labels "Alice," "Bob," and the nature of their relationship are erased before publication of the graph, the existence of the edge may provide sufficient information to re-identify Alice and Bob, as prior work, for example, that of Narayanan and Shmatikov [17], establishes.

Isomorphism-based security notions provide anonymity through indistinguishability. Their intent is to obfuscate a graph by exploiting similarity in structure within a graph, or across graphs. For example, if a graph comprises two or more distinct subgraphs that are isomorphic, erasing labels and then publishing the graph may be safe. Because, for every vertex and edge within one of those subgraphs, there is at least one other vertex and edge from a different subgraph that is identical to it in all respects. This provides anonymity for both vertices and edges. In other words, the labels of these vertices and edges cannot be re-discovered from the information available.

A powerful property of isomorphism-based approaches to security is that they can provide unconditional security, i.e. the graph is secure even if the attacker has unlimited computing resources. For example, in Figure 1.1, Graph (a) has two vertices each with degree 1, and they are indistinguishable in every way. Therefore an attacker is unable to tell the two vertices apart, notwithstanding the computational power she has. Indeed, some prior work intentionally makes the attacker stronger by providing her more information. For example, the work of Imeson et al. [10] provides the attacker with two graphs and it is known that one is a subgraph of the other. As another example, the work of Zhou and Pei [27] assumes that the attacker has complete knowledge of an entire 'neighbourhood' around some vertices.

Security for RDF Our interest in the security of graphs is motivated by our interest in securing RDF [6]. RDF is a graph-based data model designed for the web. It is used in large graph databases such as StarDog [11] and Apache Jena TBD [13]. Examples of use of RDF to store data are DBPedia [2] and GovTrack [5]. We formally define RDF graphs below.

Definition 1. *(RDF Graph) An RDF Graph G is a set of triples of the form $\langle s, p, o\rangle$, where s, p are* Internationalized Resource Identifier (IRI) *or* Blank Symbol*, o is a* IRI, Literal *or* Blank Symbol*. The component s is called a subject, p is called a predicate, and o is called an object.*

An *IRI* is a generalization of Universal Resource Identifier (URI) that allows more Unicode characters. It is used in RDF as a globally unique identifier for an entity. It

```
@prefix   people: <http://www.people.com/>.
people:amy   people:gender   female.
people:amy   people:age   22.
_:b   people:gender   female.
_:b   people:age   30.
```

Figure 1.2: An RDF graph is a set of triples

explicitly specifies which entity the vertex represents. In other words every appearance of the same IRI represents the same entity even in different RDF graphs. A *blank symbol* serves the same function as an IRI, except that it is anonymous, i.e. does not reveal what entity it represents. On the other hand, a *Literal* in RDF is similar to that in programming languages. They are syntactic representations of boolean, character, string, or numeric values.

A triple in RDF can be interpreted as a vertex-edge-vertex relationship, and an RDF graph is naturally represented as a directed graph with labels on both vertices and edges. Figure 1.2 is an example of an RDF graph. In Figure 1.2, *people:amy* is a shorthand for *http://www.people.com/amy*, which is an IRI. If the same IRI appears, even in another RDF graph, it denotes the same entity. Similarly *people:gender* and *people:age* are IRIs. *_:b* on the other hand denotes a blank node, with a random identifier *b*. The edge in an RDF triple is sometimes called a *predicate*, and in the official RDF Semantics [9], a predicate in an RDF graph can be only be IRI. However, ter Horst [22] introduced a notion of *Generalized RDF Graph* which allows blank symbolsx to be used as the predicate in a triple. In this work, we will use the generalized notion of RDF graph. Our result however is applicable to the more restricted notion too.

An appealing feature of representing and storing data as RDF is that RDF has a precise and meaningful semantics [9, 22]. The RDF semantics provides a formal specification of when truth is preserved by transformations of RDF or operations which derive RDF content from other RDF, which is meaningful even in the context of security as we will show in Chapter 3. Another attraction with RDF is that it comes with a de facto standard query language: SPARQL[18], a recursive acronym for *SPARQL Protocol and RDF Query Language*.

In this work, we generally present the RDF graph as literally a graph, which is a widely accepted way of perceiving and rendering RDF. The graph that corresponds to the triples in Figure 1.2 is shown in in 1.3. In Figure 1.4 we show a query in SPARQL, and the result

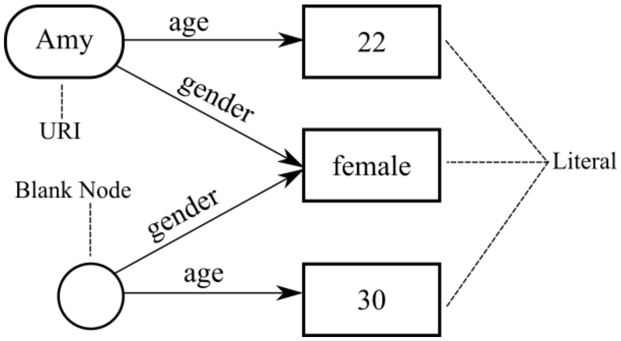

Figure 1.3: Visualization of the RDF graph in 1.2

```
    SELECT ?x ?y{
      WHERE ?x age ?y .
      FILTER (?y < 25)
    }
```

Figure 1.4: Example SPARQL query for the graph shown in Figure 1.3, the query returns "Amy 22"

of that query when it is issued against the RDF graph in 1.3. The meaning of the query is straightforward: return all x and y such that *"x age y"* is a triple, and y is less than 25. We see that the SPARQL syntax is similar to that of SQL. A variable in a SPARQL query is denoted by a preceding question mark. x and y are variables in the example.

There is a considerable volume of work on securing RDF (see, for example, [12, 20, 19]). In our assessment, there are some important gaps in such existing work. One is that prior work lacks a precise notion of security. For example, works such as [12, 19] considered only operations and techniques of sanitizing RDF, but not what the result of the sanitization should be and why the result is secure.

Another gap in existing work, in our assessment, is that none deals adequately with RDF semantics. Some pieces of work on securing RDF treat its syntactic aspects only. When a piece of work does address semantics, the approach does not seem to meaningfully leverage or reconcile it. Rather the approach seems to be somehow retrofit a proposed approach for securing RDF to semantics. Certainly none of the existing pieces of work, to our knowledge, addresses the manner in which SPARQL may behave when querying data that is secured as proposed in the work.

Threat model Below is the attack model that we consider. The data holder has a graph G and a set of vertices A from G that needs to be anonymized. Let G' be the anonymous version of G released to public.

We model the adversary's prior knowledge with another graph H. H is supposedly a subgraph of G, but the exact detail of H is unknown to the owner of G.

Since H represents the information the adversary collected through other channels, H is independent from G', the released graph. However the adversary could potentially combine H and G' to discover more information about G. For example certain vertices that are anonymous in G' might not be so in H. If the adversary could reveal the label of an anonymous vertex in G' by mapping some vertices in H to some vertices in G', he/she will be able to discover more information about that vertex than is desired. In this model the attacker is assumed to have unbounded computational power.

Isomorphism-based security provides anonymity by makeing sure that for every vertex v there are several other ones that are indistinguishable from v, with all the information available. This lowers the chance that v is correctly re-identified, because the attacker would have to make a blind guess from the possible options. Let L be the event that any vertices in A are re-identified. The released graph G' is *k-secure* if $Pr(L|H) \leq \frac{1}{k}$ for any H. This notion requires every vertex in A to be in-differentiable from at least $k-1$ other vertices in G' regardless of the adversary's prior knowledge. Note that the vertices in A does not even need to be present in G'. For example if we remove the presence of all vertices from A in G', G' would be *k-secure* as $Pr(L|H) = 0$.

Besides the notions that provide security through indistinguishability, there exists another mainstream notion of data security, which is called differential privacy [7]. However, differential privacy focuses on providing accurate statistics of the data, without revealing the original data, whereas in our model, the entire graph is released, and the security notions does not need to guarantee the accuracy of summary statistics.

Our contributions We make two sets of contributions. The first comprises broad contributions to isomorphism-based approaches to securing graphs. The second is to securing RDF in a manner that preserves semantics.

Contribution 1 We first assess, in a unified way, four seemingly different notions of isomorphism-based security in the context of graphs, that have been proposed in two different application domains. We first pose and answer the following question: is a particular notion stronger than another? Our characterization of stronger is precise. A notion is at least as strong as another if satisfaction of the former implies satisfaction of the latter. It is stronger if the converse is not necessarily true. We show that indeed, it is possible to "layer" the four notions in this regard (see Figure 2.1 in Chapter 2).

Then, we pose and answer a natural follow-up question: is strength of security correlated to computational complexity? That is, suppose a notion of security, N, is stronger than another notion, M. Is determining whether a graph satisfies N at least as computationally hard as determining whether it satisfies M? One might expect the stronger security notion to be harder to decide. We discover somewhat surprisingly however, that it is not be the case.

For example, as we show in Figure 2.3 in Chapter 2, the notion k-Symmetry is stronger than k-Subgraph-Iso. Yet, determining whether a graph satisfies the former is **ISO**-complete, and the latter is **NP**-complete. **ISO** is the set of problems that have polynomial time reductions to the problem of Iso. A problem is **ISO**-hard if it can be reduced to from Iso. **ISO**-complete is the class of problems that are in **ISO** and **ISO**-hard. **NP** is the class of decision problems for which there exist non-deterministic polynomial-time algorithms. It is known that $\textbf{ISO} \subseteq \textbf{NP}$, and believed widely that the inclusion is strict [16]. That is, a problem that is **ISO**-complete is believed to be strictly easier, in the worst-case, than a problem that is **NP**-complete.

These findings are important because they help us choose an appropriate notion of isomorphism-based security when information is represented as a graph. The two distinct axes, strength and computational hardness, are important considerations in the choice. In this work, for RDF, we pick the two strongest notions: *k-symmetry* and *k-automorphism*.

Contribution 2 We adapt *k-symmetry* and *k-automorphism* to RDF. This adaptation is a generalization because an RDF graph generalizes a graph. In the context of RDF, we address the problem of preserving as much information as possible while making an RDF graph secure. For this purpose we propose that the secured graph should be entailed by the original graph. The exact definition of entailment is discussed in Chapter 3, but informally the entailment relationship ensure that the secured graph is semantically consistent with the original graph, e.g. the information of the secured graph does not deviate from the original graph. Moreover we measure the cost of security quantitatively by the distance between original graph and secured graph on the lattice induced by entailment. This measurement allows us to minimize the cost, while ensuring security.

With the information taken into consideration, we then address the problem of securing an RDF graph so it possesses a chosen security property. We prove that achieving k-*automorphism* with the information constraint in RDF is **NP**-complete, and provide a CNF-SAT encoding for the problem.

Organization The remainder of our paper is organized as follows. In Chapter 2 we conduct a comparative study over isomorphism-based security notions. The security notions of our choice are then adapted for RDF in Chapter 3, where we also address the matter of preserving information while achieving security in the context of RDF. The discussion in Chapter 3 results in a CNF-SAT encoding, which is used to evaluate the effectiveness of our security notions in Chapter 4. Prior works are discussed in Chapter 5.

Chapter 2

Isomorphism-Based Security

In this chapter, we characterize the four different isomorphism-based notions of security of which we are aware, and that we consider in this work. These notions have been proposed in prior work in two different contexts: social networks[24, 27, 28], and the security of digital ICs [10].

In Chapter 1, we introduce graph isomorphism, automorphism and subgraph isomorphism. For example, in Figure 1.1, Graph (a) is (non-trivially) automorphic, and it is subgraph-isomorphic to Graph (b). Given these basic characterizations, we now define the properties that underlie the four notions of security we consider. We begin with *k-subgraph-isomorphism*.

Definition 2. *(k-subgraph-isomorphism) Given $\langle H, G, k\rangle$, where H, G are graphs with G a sub-graph of H, and k an integer, H is* k-subgraph-isomorphic *to G if and only if for each $v_i \in G$ there exist k different subgraph isomorphisms $f_{i,1}, \ldots, f_{i,k}$ from H to G such that $f^{-1}_{i,m}(v_i) \neq f^{-1}_{i,n}(v_i)$ if $m \neq n$.*

Subgraph isomorphisms are partial functions from the set of vertices in H to the set of vertices in G. That is, a subgraph isomorphism f maps only a subgraph of H to G, and therefore f may not be defined for all vertices in H. However for every vertex v in G, there must be a vertex v' that $f(v') = v$ by definition of isomorphism. That is the reason why we use f^{-1} in the above definition.

As the above definition mentions two graphs, H and G, and in our threat model that we discuss in Chapter 1 we discuss the protection of one graph only, it is important to point out the manner in which *k-subgraph-isomorphism* has been employed for security.

The intent of Imeson et al. [10] is that G is known to be a subgraph of H and the asset they seek to secure. Although H and G are both available to the attacker, H is perceived to be fixed and publicly known, therefore it's only possible to enforce a security property on G. Why they model it this way is specific to the application that Imeson et al. [10] consider and we refer the reader to that work for the details. Our point here is that when we discuss what we call the strength of security in the next section it is important to identify the graph that we seek to protect.

If we take *k-subgraph-isomorphism* out of its original context and look at it under our threat model discussed in Chapter 1, it provides a strong security guarantee: even if the adversary's prior knowledge H is a superset of the released graph G, the anonymous vertices in G still cannot be uniquely identified.

Next, we define what we call *k-symmetry* and *k-automorphism*. In this regard, some clarification of the terms is needed. The *k-automorphism* notion, to our knowledge, was first proposed by Zou et al. [28]. In that work, after proposing a notion of security that they call *k-automorphism*, they attach to it an additional condition that they call the "different match principle". Their work deals exclusively with *k-automorphism* with the different match principle.

The property of *k-symmetry* is from the work of Wu et al. [24]. That work simply drops the qualification "with the different match principle" in referring to the work of Zou et al. [28]. We adopt the approach of Wu et al. [24], and simply call the property *k-automorphism*, without qualification. We discuss the distinction between *k-symmetry* and *k-automorphism* once we define the two.

Definition 3. *(k-symmetry [24]) Given $\langle G, k\rangle$ where G is a graph and k an integer, G is k-symmetric if and only if for each vertex v_i in G there exist $k-1$ different automorphisms $f_{i,1}, ..., f_{i,k-1}$ such that (a) $f_{i,m}(v_i) \neq f_{i,n}(v_i)$ if $m \neq n$, and (b) $f_{i,j}(v_i) \neq v_i$.*

Definition 4. *(k-automorphism [28]): Given $\langle G, k\rangle$ where G is a graph and k an integer, a graph G is k-automorphic if and only if there exist $k-1$ different automorphisms $f_1, ..., f_{k-1}$ for G such that the following two properties are satisfied by all $v \in G$: (a) $f_i(v) \neq f_j(v)$ for all distinct pairs i, j, and, (b) $f_i(v) \neq v$ for all i.*

The difference between *k-symmetry* and *k-automorphism* is somewhat nuanced. In the former, it suffices that for each vertex, k different functions exist. Thus, the k functions may be different for each vertex. In *k-automorphism*, on the other hand, the same set of k functions must apply to all vertices. Certainly, if a graph is *k-automorphic*, it is *k-symmetric*. Wu et al. [24] articulate the question as to whether the converse is true, and leave it open. Addressing that question is beyond the scope of our work. In Lemma 4 of Section 2.1 we provide some evidences that a graph that is *k-automorphic* is not necessarily *k-symmetric*. However it should be noted that we are unaware of any graph that is *k-symmetric*, but not *k-automorphic*.

In Figure 1.1, both Graphs (a) and (b) are *2-automorphic* and *2-symmetric*. Indeed, Graph (b) is *4-automorphic* and *4-symmetric*. However, to demonstrate *2-symmetry* for Graph (a), there exist different sets of functions for the different vertices. That is, the set of functions we use to establish *2-automorphism* is not the only set that is available to establish *2-symmetry*.

Specifically, in Graph (b) in Figure 1.1, suppose we name the two vertices of degree 1, a and b. And the two vertices of degree two, c and d. Then, the identity mapping, and the automorphism that maps a to b, b to a, c to d, and d to c establish that Graph (b) is *2-automorphic*. In addition to these functions, the function that maps a and b to themselves, and c and d to one another, may be used as one of the functions to establish *2-symmetry* for c, d.

In summary, there do appear to be some differences between *k-symmetry* and *k-automorphism*. However, whether these differences are of any consequence in our context is an open question. We point out, in the next section on strength of security, that a resolution to this question may have implications to the strictness of containment of Iso in **NP**.

We now define the property that underlies the fourth isomorphism-based notions we consider. It is *k-subgraph-isomorphism*.

Definition 5. *(k-neighbourhood-isomorphism [27]) Given a graph $G = (V, E)$ and integers k, we say a vertex $u \in V$ is* k-neighbourhood-isomorphic *in G if there are $k - 1$ other vertices $v_1, ..., v_{k-1} \in G$ such that $neighbour(v_1), \ldots, neighbour(v_{k-1})$ and $neighbour(u)$ are isomorphic, where $neighbour(v)$ is the subgraph of G induced by the set of vertices within distance 1 to v. The graph G is* k-neighbourhood-isomorphic *if every vertex in G is* k-neighbourhood-isomorphic.

The function $neighbour : V \longrightarrow G$ used in the above definition denotes what is called the d-neighbourhood of vertex v in [27]. The d-neighbourhood of a vertex v in a graph is the subgraph of G that contains all vertices within distance d to v, including v itself, and all edges in G that connect these vertices. The work of Zhou and Pei [27] is focused on 1-neighbourhood only, therefore we will follow their conventions, and assume $d = 1$ unless otherwise specified. Moreover, what we call *k-neighbourhood-isomorphism* is called *k-anonymity* in their work. We adopt our nomenclature to more clearly identify the notion, and because the term *k-anonymity* has been used extensively in the altogether different context of relational tables.

In Figure 1.1, Graph (c) is 3-*neighbourhood-isomorphic*. That is, if we look at the subgraph within distance 1 of each vertex, there exist two other subgraphs to which it can map. However the graph is not *2-symmetric* because the rightmost two vertices cannot be mapped to anything but themselves in an automorphism. Note that *k-neighbourhood-isomorphism* imposes a stricter restriction than merely considering the degree of a vertex. We refer the reader to the work of Zhou and Pei [27] for a more detailed discussion.

2.1 Relative Strength of Security

We now compare the four notions we introduce in the previous sections from the standpoint of what we call strength of security. What we mean by strength of security is the following. Suppose a graph G satisfies one of the four notions, N, does it necessarily satisfy another, M? That is, logical implication. If so, we say that the property N is at least as strong as M. We write this as $N \implies M$. If the converse is not necessarily true, We write this as $M \not\Longrightarrow N$, and say that N is stronger than M.

The reason such a comparison is meaningful, is that in our context, it helps us choose a particular notion as the underlying notion of security. Indeed, in our application to RDF (see Chapter 3), part of our reason for the choice of *k-symmetry* and *k-automorphism* is based on the results we present in this section on strength of security.

To our knowledge, no prior work places all four of these isomorphism-based notions of security in the same context. Therefore, to our knowledge, the results in this section are novel. Figure 2.1 summarizes the results we establish in this section.

k-symmetry* vs. *k-subgraph-isomorphism We first show that *k-symmetry* is at least as strong *k-subgraph-isomorphism*, and strictly so. In this context, we recall our discussion from the last section on the graph that is the asset to be protected in *k-subgraph-isomorphism*. Because the notion of *k-subgraph-isomorphism* mentions two graphs: H and G. The asset is G. Therefore, in the following theorem, we instantiate the same graph G for both *k-symmetry* and *k-subgraph-isomorphism*, and the same value k. We then show that if G is *k-symmetric*, then for every H that is a supergraph of G, H and G must be *k-subgraph-isomorphic*. This is a strong guarantee in the context of *k-subgraph-isomorphism* as we promise that the graph is secure no matter what super-graph of G is known to the attacker.

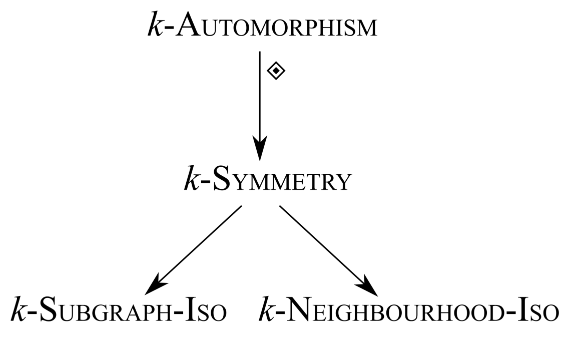

Figure 2.1: Strength of Security. The picture expresses that *k-automorphism* is stronger than *k-symmetry*. The diamond denotes our conjecture that this relationship is strict. *k-symmetry* in turn is strictly stronger than each of *k-subgraph-isomorphism* and *k-neighbourhood-isomorphism*. *k-subgraph-isomorphism* and *k-neighbourhood-isomorphism* are incomparable to one another.

Lemma 1. *k-symmetry* $\Longrightarrow$ *k-subgraph-isomorphism.*

Proof. If a graph G is *k-symmetric*, then for each vertex $v_i \in G$ there are $k-1$ isomorphisms $f_{i,1}, ..., f_{i,k-1}$ from G to itself such that $f_{i,m}(v_i) \neq f_{i,n}(v_i)$ when $m \neq n$, and $f_{i,j}(v_i) \neq v_i$ for all i, j. Given any super-graph H of G, we know that there is a subgraph G' of H that is isomorphic to G. Let the isomorphism from G to G' be g, then $g(f_{i,m}(v_i)) \neq g(f_{i,n}(v_i))$ when $m \neq n$ because g is a bijection, and $f_{i,m}(v_i) \neq f_{i,n}(v_i)$.

For each $f_{i,j}$, define a function $h_{i,j} = (g \circ f_{i,j})^{-1}$. For each v_i, there exist k subgraph isomorphism $g^{-1}, h_{i,1}, \ldots, h_{i,k-1}$ that satisfy the definition of *k-subgraph-isomorphism.* Therefore given any super-graph H of G, H is *k-subgraph-isomorphic* to G.

In other words, if a graph G is *k-symmetric*, then it is *k-subgraph-isomorphic* to itself, in which case any supergraph H of G is also *k-subgraph-isomorphic* to G.

□

Lemma 2. *k-subgraph-isomorphism* $\not\Longrightarrow$ *k-symmetry.*

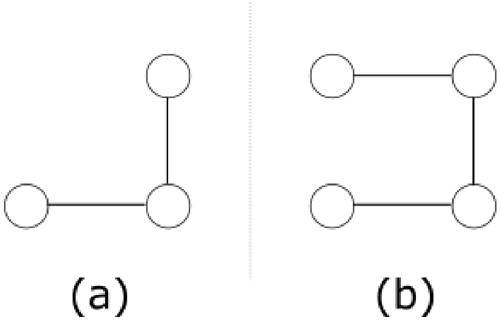

Figure 2.2: Example showing that *k-subgraph-isomorphism* does not imply *k-symmetry*.

Proof. See Figure 2.2 for a counter example. Although (b) is 2-subgraph-isomorphic to (a), (a) itself is not 2-automorphic because the vertex in the lower left corner cannot be mapped to another vertex under any automorphisms. □

k-symmetry* vs. *k-automorphism We have already mentioned that *k-automorphism* implies *k-symmetry* when defining these notions.

Lemma 3. *k-automorphism* $\implies$ *k-symmetry.*

Before we continue to discuss the reverse, we introduce the following problems as languages.

Automorphism With 1 Restriction [16] = $\{\langle G, v\rangle$ where $G = (V, E)$ and $v \in V$: G has an automorphism f such that $f(v) \neq v.\}$

Fixed-Point-Free Automorphism [16] = $\{\langle G, v\rangle$ where $G = (V, E)$ and $v \in V$: G has an automorphism such that $\forall v \in V$, $f(v) \neq v$.

Automorphism With 1 Restriction was proved to be **ISO**-complete, and Fixed-Point-Free Automorphism **NP**-complete by Lubiw [16]. We also use 2-Symmetry (respectively 2-Automorphism) to denote the problem of deciding whether a graph G is *2-symmetric* (respectively *2-automorphic*).

If a graph is *k-automorphic* then it is also *k-symmetric.* At least when $k = 2$, there are evidences suggesting the other direction is not true.

Conjecture 4. *k-symmetry* $\not\Longrightarrow$ *k-automorphism.*

We can show that there exists a Cook reduction from 2-Symmetry to Automorphism With 1 Restriction: given graph $G = (V, E)$ as input for 2-Symmetry, we simply solve Automorphism With 1 Restriction with the input $\langle G, v\rangle$ for each vertex $v \in V$. We have $G \in$ 2-Symmetry if and only if $\langle G, v\rangle \in$ Automorphism With 1 Restriction for each $v \in V$ by definition.

Because Automorphism With 1 Restriction is known to be **ISO**-complete, the above reduction means 2-Symmetry is no harder than the Iso problem. However, when 2-Automorphism is equivalent to Fixed-Point-Free Automorphism, which is known to be **NP**-complete.

We already know that *k-automorphism* $\implies$ *k-symmetry.* If *k-symmetry* $\implies$ *k-automorphism*, then 2-Symmetry is equivalent to 2-Automorphism. This would mean that Iso is in the class of **NP**-complete problems. This is the converse of common belief [16], therefore we conjecture that k-Symmetry and k-Automorphism are not equivalent.

k-symmetry vs. *k-neighbourhood-isomorphism*

Lemma 5. *k-symmetry* $\implies$ *k-neighbourhood-isomorphism.*

Proof. If G is *k-symmetric*, then for each $v \in G$ besides the identity mapping there are at least $k-1$ other automorphisms $f_1, f_2, ..., f_{k-1}$ from G to itself such that $f_m(v_i) \neq f_n(v_i)$ when $m \neq n$. In other words, the $k-1$ vertices $f_1(v_i), ..., f_{k-1}(v_i)$ would have the same neighbourhood as v_i, so v_i is *k-neighbourhood-isomorphic.* Therefore G is also *k-neighbourhood-ismorphic.* This result is valid for d-neighbourhood regardless of the value of d. □

Lemma 6. *k-neighbourhood-isomorphism* $\not\Longrightarrow$ *k-symmetry.*

Proof. See Figure 1.1 (c) for a counterexample. The graph is *3-neighbourhood-isomorphic*, however it is not even *2-symmetric* as the 2 vertices on the right side of the graph cannot be mapped to anything else but themselves in an automorphism.

□

Note that in the above theorem, if we consider d-neighbourhood where $d = n - 1$, k-Neighbourhood is the same as k-Automorphism. But this is somewhat meaningless. In particular, a key point of the work of Zhou and Pei [27] is that if a graph exhibits *k-neighbourhood-ismorphism* for small d, e.g., $d = 1$, then it yields security. As we increase

d, we naturally intuit that security does not decrease. So the key question is whether for small d, we gain security. And in this context, we seek to compare this notion of security with the others that we consider.

In the end we also point out that there is no implication relationship between *k-subgraph-isomorphism* and *k-neighbourhood-isomorphism*. For example, although (b) in Figure 2.2 is *2-subgraph-isomorphic* to (a), (a) is not *2-neighbourhood-isomorphic*. The other direction is also easy to show as (c) in Figure 1.1 is *2-neighbourhood-isomorphic* but not *2-subgraph-isomorphic* to itself.

2.2 Computational Complexity

The results from the previous section suggest that some notions of security are stronger than others. So a natural question we may ask is: is strength of security correlated with computational complexity? That is, suppose notion N is stronger than M. Is determining whether a graph G satisfies N at least as hard as determining whether it satisfies M? This question of whether a graph satisfies a notion is fundamental to any techniques we may want to devise to secure a graph.

For example, Zhou and Pei [27] and Imeson et al. [10] consider the problem of transforming a graph so that the resultant graph has a certain level of security. But an even more basic question, and indeed a lower-bound for the hardness of the problem of graph-transformation, is determining whether a given graph has a certain level of security.

This is the question we consider in this section. We ask, for the four notions of security that we consider, what the computational hardness is of determining whether a graph has the corresponding level of security.

Somewhat surprisingly, this rather basic question has not been posed in prior works, to our knowledge. Consequently, our results in this section are novel as well, to our knowledge. Our results from this section are summarized in Figure 2.3.

In this section, we abuse notation and use the mnemonics k-SUBGRAPH-ISO, k-SYMMETRY, k-AUTOMORPHISM and k-NEIGHBOURHOOD to refer to the problem of determing whether a graph G has each property. For each notion, we also assume that we are provided the additional input, k. For k-SUBGRAPH-ISO, we assume that we are also provided H.

We first observe that the problem for all the four notions is in **NP**. This establishes an upper-bound. Then, the question that remains is whether this upper-bound is tight.

Lemma 7. *k-Symmetry, k-Automorphism, k-Subgraph-Iso and k-Neighbourhood are all in **NP**.*

Proof. For all four notions, a certificate is a set of mappings. As k is at most the number of vertices in the graph, the number of mappings is at worst quadratic in the size of the graph. It is quadratic, in the worst-case, for k-Subgraph-Iso, k-Symmetry and k-Neighbourhood. It is at worst linear for k-Automorphism. A mapping is linear in the number of vertices. Thus, a certificate is efficiently sized. It can also be verified efficiently against each of the properties. □

Next we address tighter upper-bounds for two of the notions.

Lemma 8. *k-Symmetry is in **ISO**.*

Proof. We present a Cook reduction from k-Symmetry to Iso.

Given a graph G, let G' be a copy of G. For each $v \in G$, find k vertices $v' \in G'$ such that when v and v' are given the same label, we can find an isomorphism between G and G'. By definition $\langle G, k \rangle$ is in k-Symmetry if and only if we can find k such vertices $v' \in G'$ for each $v \in G$ such that $f(v) = v'$ under some isomorphism. This process, in the worst-case, requires $O(n^2)$ invocations of an Iso oracle. Therefore it is a Cook reduction. □

Lemma 9. *k-Neighbourhood is in **ISO**.*

Proof. We provide a reduction from k-Neighbourhood to Iso.

Given $\langle G, k \rangle$ where $G = (V, E)$, let $U = V$. While U is not empty, remove a vertex v from U, find all vertices v' from U such that 1-neighbourhood of v' is isomorphic to that of v, if there does not exist at least $k - 1$ other vertices with isomorphic neighbourhoods, then return *False*; Otherwise remove v and all vertices with neighbourhood isomorphic to v from U. If U is empty return *True*.

If the above algorithm returns True, then $\langle G, k \rangle \in k$-Neighbourhood, because every vertex would have $k-1$ other vertices with isomorphic neighbourhoods. Otherwise $\langle G, k \rangle \notin k$-Neighbourhood. Computing *1-neighbourhood* takes time $O(|V| + |E|)$ and needs to be done for $|V|$ vertices. Comparison of neighbourhoods requires $O(n^2)$ calls to the Iso oracle. □

Thus, under the customary assumption that **ISO** $\subset$ **NP**, we know that k-Symmetry and k-Neighbourhood are not **NP**-complete. We now consider lower-bounds for the computational hardness of each problem.

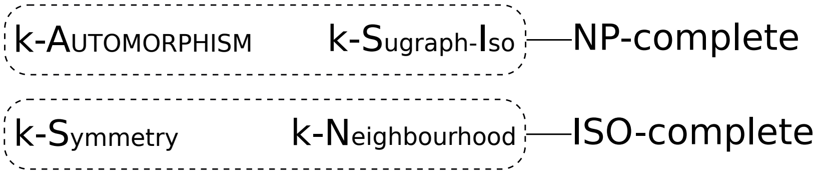

Figure 2.3: Computational Complexity for the problem of whether a graph satisfies the four notions of security we consider. If we adopt the customary premise that **ISO** $\subset$ **NP**, then k-Symmetry and k-Neighbourhood are strictly easier than k-Automorphism and k-Subgraph-Iso.

Lemma 10. *k-Subgraph-Iso is **NP**-hard.*

Proof. There exists a reduction from the Clique problem: Given a graph $G = (V, E)$ and a number k, decide if the graph contains a clique of at least size k.

Given $\langle G, k\rangle$, we create a clique of size k, and call it C. Let graph $H = G \cup C$. Solve k-Subgraph-Iso for the input $\langle H, C, k+1\rangle$, and return whatever the Oracle returns. We know that H is at least k-subgraph-isomorphic to C, because it contains a copy of C. However for H to be $(k+1)$-subgraph-isomorphic to C, G must also contain a copy of C, in which case G contains a clique of size k.

□

Lemma 11. *k-Automorphism is **NP**-hard.*

Proof. As mentioned in Section 2.1, there exists a straightforward reduction from the Fixed-Point-Free Automorphism problem introduced in Section 2.1. When $k = 2$, k-Automorphism is equivalent to Fixed-Point-Free Automorphism. Therefore k-Automorphism is **NP**-hard. □

Theorem 1. *k-Symmetry is **ISO**-hard.*

Proof. We present a Cook reduction from Iso to k-Symmetry.

Given two graphs G, H, if one of G, H is connected and the other is not, immediately return *False*. If G (respectively H) is not connected, let G (respectively H) be its complement graph. This is to ensure that the graphs are connected.

Uniquely label a vertex v from G. Give each vertex v' from H the same label in turn, and solve k-Symmetry for the input $\langle G \cup H, 2\rangle$. The oracle will return True at least once if and only if $\langle G, H\rangle \in$ Iso. This is because if G and H are isomorphic, then for every v in G, there must be a vertex v' in H that corresponds to v through an isomorphism. This reduction takes $O(n)$ calls to the k-Symmetry oracle.

□

Lemma 12. *k-Neighbourhood is* ***ISO****-hard.*

Proof. We provide a Cook reduction to k-Neighbourhood from a restricted version of Iso, introduced below.

Restricted Iso $= \{\ \langle G = (V, E), G' = (V', E')\rangle$ where G and G' are both graphs with diameter 2 and radius 1: G and G' are isomorphic. $\}$

The diameter of a graph G is the longest shortest path between any two vertices in G. The eccentricity of a vertex v is the greatest distance between v and any other vertex. The radius of a graph is the minimum eccentricity of any vertex. Restricted Iso is simply the graph isomorphism problem restricted to a certain class of graphs, and it is in itself an **ISO**-complete problem [25].

To solve Restricted Iso, for each pair of vertices v, v' such that $v \in V$, $v' \in V'$ and v, v' both have eccentricity 1, give v, v' the same unique label and solve k-Neighbourhood for the input $\langle G \cup G', 2\rangle$. If $\langle G \cup G', 2\rangle \in k$-Neighbourhood when v, v' are given the same label, the 1-neighbourhood of v and v' are isomorphic. Moreover the 1-neighbourhoods of v and v' are G and G' respectively, because the vertices have eccentricity 1. Therefore $\langle G \cup G', 2\rangle \in k$-Neighbourhood when some v, v' are given the same label if and only if $\langle G, G'\rangle \in$ Restricted Iso.

The reduction is a polynomial time reduction because there are at most $|V|^2$ pairs of vertices to label, and determining the eccentricity of a vertex takes time polynomial to the size of the graph.

□

Chapter 3

Securing RDF

We now address the matter of securing RDF in a way that is consistent with its semantics.

Our threat model is introduced in Chapter 1. That is, we seek to obfuscate an RDF graph G, and publish the result G', so that the adversary is unable to re-identify the anonymous vertices in G' to gain more private information about these vertices than he already has. The prior knowledge of the adversary is also modeled by an RDF graph H. This is why we adopt the notions *k-automorphism* and *k-symmetry*, as they provide exactly this kind of security.

Two issues remain in applying the security notions in Chapter 2 to RDF. One is that RDF is more general than the kinds of graph for which *k-automorphism* and *k-symmetry* have been considered in the literature. Therefore, we need to generalize those notions to suit RDF. We do so in Section 3.1

A second issue regards data quality. Our objective is to achieve a desired level of security with as little loss of information as possible. We achieve this goal in two ways. First is that in the process of generating G' from G, we require RDF semantics to be preserved. In Section 3.2, we first introduce the sanitization operations we use to achieve security in RDF. We then justify this choice of operations with a discussion of why and how the choice preserves RDF semantics. In addition, we provide a quantitative measure the cost of achieving this semantically consistent security, and require that this cost is minimized.

3.1 Security for RDF Graphs

In this section we define the *k-symmetry* and *k-automorphism* for RDF. The original definition *k-symmetry* and *k-automorphism* could not be applied directly to RDF because they are defined on unlabeled, undirected graphs. RDF on the other hand, can be thought of as having explicit labels on both edges and vertices, except for blank vertices and edges. These labels can contain sensitive or identifying information. Therefore a notion of security for RDF must take these labels into consideration. The definition of an RDF graph has been discussed in Chapter 1. We show below how we redefine automorphism in this context.

We define the following notations to be used in definitions. Given a graph G, let $t_1(G)$ be the set of all subjects, $t_2(G)$ the set of all predicates, and $t_3(G)$ the set of all objects. We define the terms of G to be $T(G) = t_1(G) \cup t_2(G) \cup t_3(G)$. Let B be the set of blank vertices in G, then $B \subseteq T(G)$.

Definition 6. *(Automorphism for RDF) An automorphism on an RDF graph is defined as a bijection $f\colon T(G) \longrightarrow T(G)$, which has the following properties:*

1. *Given $v \in T(G)$, $f(v) = v$ if $v \notin B$; $f(v) \in B$ if $v \in B$; and*

2. *a triple $(a, b, c) \in G$ iff $(f(a), f(b), f(c)) \in G$.*

Definition 6 is different from the traditional definition of automorphism mentioned in Chapter 1. It recognizes that a non-blank vertex is globally unique, and explicitly identified by its label. Such a vertex under an automorphism cannot be mapped to any other vertex than itself. Blank vertices on the other hand do not have explicit identities. Therefore they can be mapped to other blank vertices.

Moreover, the automorphism applies to the IRIs and blank symbols no matter they serve as vertices or edges in a graph. IRI and similarly blank symbols can serve as predicates in triples, in which case they act like edges in a graph. Different appearances of the same IRI as predicates are just instances of the same relationship, therefore an IRI edge can only be mapped to an edge under the same IRI. Different appearances of the same blank symbols as predicates also denote the same relationship, except in this case the exact identity of the relationship is unspecified. Automorphism for edges can be seen as from edge type to edge type. If we map blank edge type a to blank edge type b, we must map every edge under the type a to an edge under type b. For example, in Figure 3.1 assuming all vertices and edges are blank, there is an automorphism that maps the vertex a to the vertex b in the

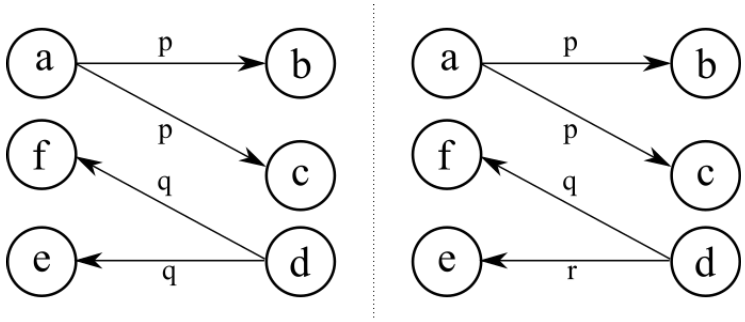

Figure 3.1: RDF automorphism

left graph, because the blank relationship p can be mapped to q. Such an automorphism however does not exist in the right graph in Figure 3.1 because the relationship p cannot be mapped to both q and r at the same time. This notion of automorphism is consistent with the RDF semantics we discuss in Section 3.2.

With the notion of automorphism settled, we then define *k-symmetry* and *k-automorphism* for RDF.

Definition 7. *(k-symmetry for RDF) Given an RDF graph G, let the anonymization set A be a set of vertices $A \subseteq t_1(G) \cup t_3(G)$, G is k-symmetric with respect to A if:*

1. *every vertex in A is blank; and*
2. *for each vertex $v \in A$, there exists k automorphisms $f_1, \ldots, f_k$ such that $f_m(v) \neq f_n(v)$ when $m \neq n$.*

Definition 8. *(k-automorphism for RDF) Given an RDF graph G, let the anonymization set A be a set of vertices $A \subseteq t_1(G) \cup t_3(G)$, G is k-automorphic with respect to A if:*

1. *every vertex in A is blank; and*
2. *There exists k automorphisms $f_1, \ldots, f_k$ such that $f_m(v) \neq f_n(v)$ for all $v \in A$ when $m \neq n$.*

In the above definitions we introduce the anonymization set A, which is used to specify the vertices that needs to be secured. The use of set A helps minimize the cost of our security. We also require A to be a subset of $t_1(G) \cup t_3(G)$ because we consider vertices to be the focus of our security notions: if a vertex remains anonymous to the adversary, even if its relationships with other vertices is known, the adversary cannot connect these relationships with their owner and could not utilise this knowledge.

3.2 RDF semantics and security

Preserving Semantics Given the above adaptations of the notions of security we consider for RDF, we return to the issue of semantics. Specifically, we seek to achieve a level of security, e.g., 4-automorphism, while preserving the semantics of the original graph. A consequence of this requirement is that it limits the mechanisms we are allowed to use to achieve security.

The notion of a blank symbol, to which we refer in the definition of RDF graph is an important one in this context, as it represents an anonymous subject, predicate or object. From a security perspective, this definition suggests a simple anonymization operation: replacing a non-blank vertex with a blank vertex. In fact this is one of the two operations we recommend for sanitizing an RDF graph.

We propose using two and only two types of operations to achieve security in an RDF Graph:

1. Removing RDF triples; and,

2. Replacing an IRI vertex or a Literal vertex with a blank vertex.

These are of course not the only operations available. Some other works we discussion in Chapter 5 employs other techniques such as adding vertices and edges. The reason we choose these two operations is not merely an intuition based on the RDF syntax, but rather to preserve RDF semantics. We introduce RDF semantics next and show that to preserve the semantics of an RDF graph only these two operations can be allowed.

An appealing feature of RDF is that it has a precise model-theoretical semantics [9]. The semantics of a language defines meanings for the constructs in the language. The RDF semantics is primarily concerned with providing a formal specification of when the truth-value of an RDF graph is preserved by transformations. The truth-value is a value indicating the relationship of an expression to truth. Take an example in natural language, the sentence “It is raining.” might be either true or false depending on the context. Similarly RDF semantics defines how we can associate a truth-value with an RDF graph and how transformation of the graph affects the truth-value. This is achieved through *RDF interpretation*. An RDF interpretation maps RDF terms such as IRIs to elements of the universe. It also maps triples and RDF graphs to truth-values, which denote whether the statements represented by the triples (or the graph that contains the triples) are true in the real world. When an RDF graph G is true under the interpretation I, we say that an I *satisfies* G. We refer the reader to [9, 22] for more details.

Another important concept that is related to RDF interpretation and important to our work is that of RDF entailment. While an interpretation decides the truth-value of an RDF graph, entailment decides whether the truth-value is preserved between different graphs. An RDF graph G is said to *entail* another RDF graph H if every interpretation that satisfies G also satisfies H. In other words, the truth of G under *any* interpretations is preserved in H. In this case we write $G \vDash H$.

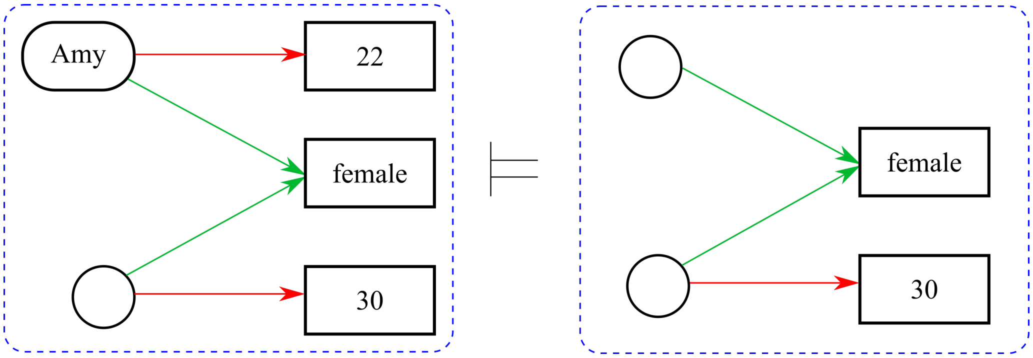

Figure 3.2: The left graph entails the right graph

From a security standpoint, this notion of entailment allows us to sanitize the graph while keeping the meanings of the graph consistent. Think of graph G as the original graph and H as the secured version to be released, if G entails H, then H will be consistent with G in semantics and therefore meanings. H might not contain as much information as G, yet all the information in H will be semantically consistent with what is in G. This is exactly the central idea of this section: to provide security to RDF graphs without disrupting the semantic consistency.

The *interpolation lemma* resulted from RDF research [9] allows us to implement this idea syntactically. To explain the interpolation lemma we first need to define the notion of *instance*: A graph G is an instance of another graph H if there exists a *partial mapping* h from the set of blank vertices in H to a set of IRIs, literals, and blank vertices such that G can be obtained from H by replacing every blank vertex b in the domain of h with $h(b)$. The interpolation lemma then simply states that an RDF graph G entails another RDF graph H if and only if a subset of G is an *instance* of H.

The interpolation lemma translates the semantic definition of entailment into simple syntactic requirements. The two operations we proposed at the beginning of this section are derived directly from this lemma. It is easy to see that any sanitized graph H produced from these operations would be entailed by the original graph G, and no other meaningful operations could be used without destroying this property.

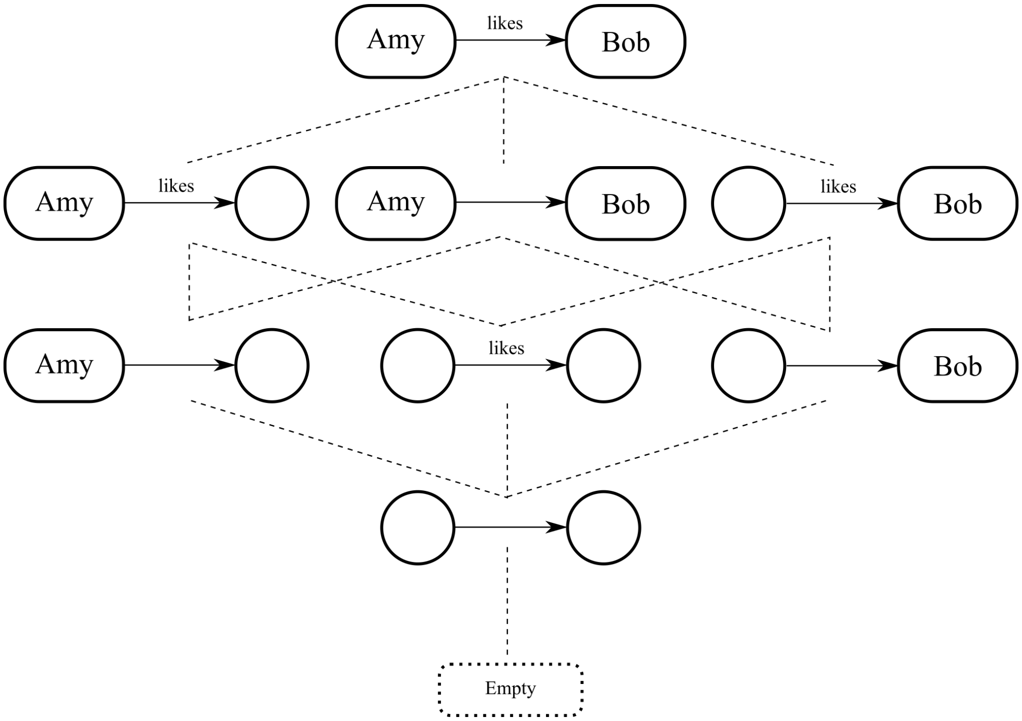

Figure 3.3: Lattice induced by entailment

Here, we provide a simple example of entailment. Consider the left graph in Figure 3.2. An operation that results in a new graph that is entailed by this original graph is blanking. We could, for example, blank the vertex labeled "Amy." Thus, the new graph still has a vertex that represents a person, and in that manner, preserves semantics. Similarly, we can delete Amy's age from the graph and still preserve the graph's semantics. Therefore the left graph entails the right graph in Figure 3.2.

As our example and discussions above suggest, entailment carefully preserves the original graph's meaning. This is exactly the intent of RDF semantics.

It should be mentioned that there is a semantic mismatch between RDF and its query language SPARQL which slightly affects entailment. In RDF two blank vertices are not assumed to be necessarily distinct, whereas in SPARQL queries two blank vertices are considered to represent distinct entities. For more detail of this matter we refer the readers to [1]. In this work we follow the SPARQL semantics as this is the one that is used in practice. This decision affects the definition of *instance*: an RDF graph G is an instance of another RDF graph H if there is an *injective* partial mapping h from the set of blank vertices in H to a set of IRIs, literals, or blank vertices that do not already exist in H, such that G can be obtained from H by replacing every blank vertex b in the domain of h with $h(b)$. In other words, this adds to the original definition of *instance* that two blank can not be replaced with the same vertex when transforming H to G.

Cost of Security Entailment ensures that the information of the RDF graph is correct, but it does not measure how much information there is in the secured graph. For example, the empty graph is entailed by every graph, but it is probably not what people would like to be released. Therefore we provide a measure of information based on entailment.

The notion of entailment induces a lattice on the all possible anonymizations of the original graph. At the top of the lattice is the original graph, and at the bottom, is the empty graph. An intermediate entry in the lattice is entailed by its ancestors in the lattice, and an entry's descendants are entailed by this entry and thus all of its ancestors.

Lemma 13. *The entailment relationship $\vDash$ is a partial order over the set of all possible anonymizations of G.*

Proof. Let P be the set of all possible anonymizations of G. Note that P contains G itself and the empty graph. We prove that the entailment relationship $\vDash$ is reflexive, transitive, and anti-symmetric.

Reflexivity: An RDF graph always entails itself.

Transitivity: If $G \vDash H, H \vDash I$, then by interpolation lemma a subset of G is an instance of H, also a subset of H is an instance of I. That means a subset of G is an instance of I. Therefore $G \vDash I$.

Antisymmetry: Given two RDF graphs G and H, If $G \vDash H$ and $H \vDash G$, then G and H are instance of each other. For blank vertices, let h be the partial function used to derive instance relationship. We define $f : T(G) \to T(H)$ where $f(v) = h(v)$ for $v \in domain(h)$, and $f(v) = v$ otherwise. The function f is a bijection because h is an injection and $h(b) \in T(G)$, as discussed in Section 3.2. Therefore the relationship $\vDash$ is antisymmetric by definition [22]. □

It should be noted that the above result is based on the SPARQL notion of entailment. It is unclear whether this result applies to the original RDF simple entailment.

Figure 3.3 shows a lattice induced from a single triple. Note that on the lattice, *the deletion of a triple can only happen after every vertex in the triple is blanked.* This is because we believe deletion of triples remove more information than blanking vertices. A triple with every vertex blanked still retains its structural information, but once the triple is deleted, nothing is left.

The lattice provides an intuitive way to measure the cost of achieving security: we simply count the distance from original graph (the top of lattice) to the secure graph H on the lattice. The longer distance, the more information we lost, and the higher the cost of security is.

Securing an RDF graph Now that we have notion(s) of secure and a clear measure of information loss, we address the problem of devising an algorithm that transforms an RDF graph G into another graph G' such that G' preserves the semantics of G but also has a desired level of security. We define the decision problem in the form of language as below.

Minimal-k-Automorphism $= \{\langle G, k, r, A\rangle$ where G is an RDF graph, $A \subseteq t_1(G) \cup t_3(G)$, and k, r are integers : there exists an RDF graph G' such that G' is k-automorphic with respect to A and G' is reachable from G in no more than r steps in the lattice induced by entailment.$\}$

The above problem is at least as hard as the k-Automorphism problem discussed in Chapter 2.

Theorem 2. Minimal-k-Automorphism *is* ***NP****-hard.*

Proof. There exists an obvious reduction from the k-Automorphism problem defined.

Given $\langle G, k\rangle$ as the input for k-Automorphism, we construct an RDF graph G' from G. For every vertex in G, we create a blank vertex in G'. We use only one blank edge type in G' for all the edges we create and for every edge in G, we create two edges in G' that points to opposite directions. Let set A include all the vertices from G'. We then solve Minimal-k-Automorphism for the input $\langle G', k, 0, A\rangle$. It is easy to see that $\langle G', k, 0, A\rangle \in$ Minimal-k-Automorphism if and only if $\langle G, k\rangle \in k$-Automorphism. □

Similarly we define a problem Minimal-k-Symmetry based on k-Symmetry. It is easy to see from a similar reduction that Minimal-k-Symmetry is **ISO**-hard.

Cnf-Sat Encoding Given the above complexity result, there is no known algorithm for solving these problems efficiently. Therefore we provide reductions from

MINIMAL-k-SYMMETRY and MINIMAL-k-AUTOMORPHISM to CNF-SAT, and use a SAT solver to solve these problems. We now discuss our reduction from MINIMAL-k-SYMMETRY to CNF-SAT.

Input: $\langle G, k, r, A \rangle$ where G is an RDF graph, k, r are integers, $k < |t_1(G) \cup t_2(G)|$, $A \subseteq t_1(G) \cup t_3(G)$.

Additional notations:

We define the following notations to be used in the encoding.

- $V = T(G)$. Let $n = |T(G)|$. $V = [1, n]$.
- V comprises two partitions: B (blank symbols) and $\overline{B}$. Assume $B = [1, b], \overline{B} = [b+1, n]$.
- Let $f_{i,j} : V \longrightarrow V$ be an automorphism. We define a mapping $g_{i,j} : G' \longrightarrow G'$ for each $f_{i,j}$ such that for $t_1, t_2 \in G'$ such that $t_1 = (s_1, p_1, o_1), t_2 = (s_2, p_2, o_2)$,

$$g_{i,j}(t_1) = t_2 \iff f(s_1) = s_2 \land f(p_1) = p_2 \land f(o_1) = o_2. \tag{3.1}$$

Boolean variables:

We create the following boolean variables in our CNF-SAT encoding.

1. $\forall v \in V$, a variable x_v such that $x_v = 1$ iff v is blank(ed).
2. $\forall t \in G$, a variable y_t such that $y_t = 1$ iff t is deleted.
3. $\forall v \in V$, a variable z_v such that $z_v = 1$ iff all tuples in which v appears are deleted.
4. $\forall \alpha, \beta \in V, \gamma \in A, \forall \delta \in [1, k]$, a variable $f_{\alpha,\beta,\gamma,\delta}$ such that $f_{\alpha,\beta,\gamma,\delta} = 1$ iff $f_{\gamma,\delta}(\alpha) = \beta$.
5. $\forall \alpha, \beta, \in V, \gamma \in A, \forall \delta \in [1, k]$, a intermediate variable $b_{\alpha,\beta,\gamma,\delta} \iff f_{\alpha,\beta,\gamma,\delta} \land x_\beta \land \neg z_\beta$.
6. $\forall t_1, t_2 \in G, \forall \gamma \in A, \forall \delta \in [1, k]$, a variable $g_{t_1,t_2,\gamma,\delta}$ such that $g_{t_1,t_2,\gamma,\delta} = 1$ iff $g_{\gamma,\delta}(t_1) = t_2$.
7. $\forall t_1, t_2 \in G, \forall \gamma \in A, \forall \delta \in [1, k]$, an intermediate variable $h_{t_1,t_2,\gamma,\delta} \iff g_{t_1,t_2,\gamma,\delta} \land \neg y_{t_1}$.

Clauses, i.e., constraints:

Below we list the CNF-SAT clauses that we generate from the input. For the sake of brevity and clarity we write logical implications of the form $a \implies b$ where necessary. Clauses of these types are transformed to $\neg a \land b$ in the actual implementation.

1. $\forall v \in B$, a clause: x_v.

   This encodes vertices already known to be blank.

2. $\forall t = \langle s, p, o \rangle \in G$: $y_t \implies x_s \wedge x_p \wedge x_o$.

   In the lattice, deletion of a triple occurs only after all vertices in the triple are blank(ed).

3. $\forall v \in V$, let $T_v = \{t : t = \langle s, p, o \rangle \in G \wedge (v = s \vee v = p \vee v = o)\}$. $\forall v$, a clause: $y_{t_{v_1}} \wedge \ldots \wedge y_{t_{v_{|T_v|}}} \iff z_v$.

   Vertex considered deleted if and only if every triple in which it appears is deleted.

4. clauses that correspond to: $x_{b+1} + \ldots x_n + y_{t_1} + \ldots + y_{t_{|T|}} \leq r$.
   Number of triples from G that are deleted + Number of vertices that are originally not blank being blanked $\leq r$.

5. $\forall i \in V, \forall \gamma, \delta$: $\neg x_i \wedge \neg z_i \implies f_{i,i,\gamma,\delta}$.

   Vertex not blank and not deleted only if it is mapped to itself.

6. $\forall i \in V, \forall \gamma, \delta$: $x_i \wedge \neg z_i \implies (f_{i,1,\gamma,\delta} \wedge x_1 \wedge \neg z_1) \vee \ldots \vee (f_{i,n,\gamma,\delta} \wedge x_n \wedge \neg z_n)$.

   Vertex is blank and not deleted only if it is mapped to at least one blank vertex that has not been deleted.

   - This is transformed to
     $x_i \wedge \neg z_i \implies b_{i,1,\gamma,\delta} \vee \ldots \vee b_{i,n,\gamma,\delta}$, and
     $b_{\alpha,\beta,\gamma,\delta} \iff f_{\alpha,\beta,\gamma,\delta} \wedge x_\beta \wedge \neg z_\beta$

7. $\forall i, j, p \in V, i \neq j, \gamma \in A, \delta \in [1, k], \neg f_{i,p,\gamma,\delta} \vee \neg f_{j,p,\gamma,\delta}$.

   Two vertices cannot both be mapped to the same vertex.

8. $\forall t_1 = (s_1, p_1, o_1), t_2 = (s_2, p_2, o_2) \in G$, $\forall \gamma \in A, \delta \in [1, k]$,

   - $\neg y_{t_1} \wedge \neg y_{t_2} \wedge g_{t_1,t_2,\gamma,\delta} \implies f_{s_1,s_2,\gamma,\delta} \wedge f_{p_1,p_2,\gamma,\delta} \wedge f_{o_1,o_2,\gamma,\delta}$
   - $\neg y_{t_1} \wedge \neg y_{t_2} \wedge f_{s_1,s_2,\gamma,\delta} \wedge f_{p_1,p_2,\gamma,\delta} \wedge f_{o_1,o_2,\gamma,\delta} \implies g_{t_1,t_2,\gamma,\delta}$

   We perceive triples as edges in automorphism. This constraint encodes the definition of function $g_{\gamma,\delta}$.

9. $\forall t \in G$, $\forall \gamma \in A, \delta \in [1, k]$,
   $\neg y_t \implies (g_{t,t_1,\gamma,\delta} \wedge \neg y_{t_1}) \vee \ldots \vee (g_{t,t_{|G|},\gamma,\delta} \wedge \neg y_{t_{|G|}})$.

- This is transformed to
  $\neg y_t \implies h_{t,t_1,\gamma,\delta} \wedge \ldots h_{t,t_{|G|},\gamma,\delta}$, and
  $h_{t,t_i,\gamma,\delta} \iff g_{t,t_i,\gamma,\delta} \wedge \neg y_{t_i}$.

10. $\forall t, t_1, t_2 \in G$ such that $t_1 \neq t_2$, $\forall \gamma \in A, \delta \in [1,k]$, $\neg g_{t,t_1,\gamma,\delta} \vee \neg g_{t,t2,\gamma,\delta}$

    The three constraints above ensures that natural constraint of the automorphism: there is a one-to-one mapping between the triples that are not deleted.

11. $\forall i \in A, \forall j \in V, \forall$ distinct δ, ψ: $\neg z_i \implies \neg d_{i,j,i,\delta} \vee \neg d_{i,j,i,\psi}$.

    *The "k" different mappings in k-*SYMMETRY*, defined only for vertices that have not been deleted.*

A similar reduction can be constructed from k-AUTOMORPHISM to CNF-SAT. The only change is that where a variable $f_{\alpha,\beta,\gamma,\delta}$ is used, we use $f_{\alpha,\beta,\delta}$ to encode only k automorphisms in total, instead of k automorphisms for each $v \in A$.

Chapter 4

Empirical Results

We have conducted an empirical assessment of *k-symmetry* and *k-automorphism* on RDF with the encoding provided in Section 3. Our intent with the empirical study is to investigate the relationship between security and information under realistic graphs of different structures. More specifically we would like to understand, in practice, how the various parameters such as the input graph G, the level of security k, the anonymization set A and the minimum cost r interact. For instance, given G and A, it seems reasonable to expect that the minimum cost r required grows with k, but how fast does it grow? Is it harder to anonymize a larger graph? These are some of the questions we investigate in this section. We intentionally keep our RDF graph inputs small so it is easier to intuit relationships between the parameters.

It should be noted that although we have investigated both *k-symmetry* and *k-automorphism*, these two notions exhibit the same behaviour during our experimentations. In other words, achieving these two security notions for the same k has the same cost for all the inputs that were tested. Therefore we do not differentiate them in the rest

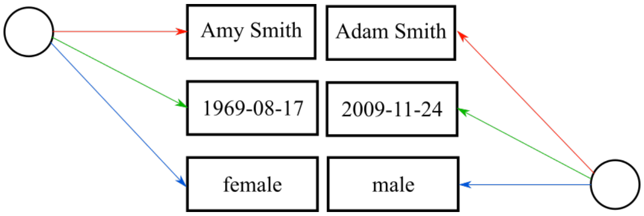

Figure 4.1: Attribute graph. The red edges denote the “full name” predicate. Green edges denote “birthday” and blue “gender”.

of this section, and just refer to them as *k-security*.

We conduct our empirical research over two types of graphs. The first type, which we call *network* graphs, are based on examples used in social network anonymization such as [27]. The second, which we call *attribute* graphs are information of US senators retrieved from [5]. The attribute graphs are real life examples of RDF data published online. It also closely resembles other online RDF data sources such as DBpedia in structure. Figure 4.1 is an example of attribute graphs, and Figure 4.2 of network graphs. Note that there are several types of edges in Figure 4.1, so we use different colours to denote the different edge types. On the other hand, every edge in Figure 4.2 is meant to denote the *knows* relationship, therefore they are all of the same colour.

The reason we have chosen these two types of graphs for experiments is that they highlight two typical, yet distinct ways of utilizing RDF graphs. The *network* graphs highlights the relationship between vertices. All the vertices in a *network* graph represent the same type of entities (people in this example) and the graph describes relationship between the entities. The *attribute* graphs, on the other hand, describes attributes of entities. Some of the vertices in the graphs represent people while others represent attributes of these people such as name, gender, and birthday. The *attribute* graph does not contain relationship between entities of the same type. These two characteristics can be mixed of course in one graph. However in practical RDF graphs they are often separated, and from a theoretical standpoint it would be interesting to investigate how their differences would affect security.

As is shown in Figure 4.1, the attribute graphs have a lot of small components of similar structures. In fact our empirical result shows that this property of the attribute graphs makes it easier to achieve higher-level of security. As we can see from Figure 4.3, for the input attribute graph, achieving 3-security and 5-security has the same cost. In other words, when 3-security is achieved, 4-security and 5-security is also achieved. This is not a coincidence, but a result of the structure of the attribute graph.

The network graph, in contrast, does not possess this property. As is shown in Figure 4.4, the cost r increases steadily with the security parameter k. In this case, the cost only stops increasing after the entire graph is removed, in which case it can be seen as infinitely secure. This suggests that achieving the same level of security in an attribute graph is easier than in a network graph of similar size. The structural randomness of the network graph makes it harder to secure.

We also looked at the relationship between the size of the graph and the cost of security. Figure 4.7 and Figure 4.6 shows that when the network graphs becomes bigger (with more vertices and proportionally more edges), even if the number of vertices to be secured remains the same, it becomes more costly to achieve a certain level of security. Figure 4.7

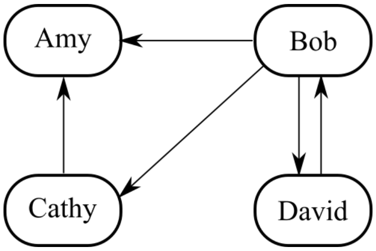

Figure 4.2: Network graph

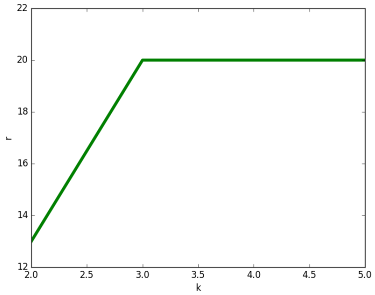

Figure 4.3: Relationship between k and r for attribute graphs

shows the case where we fix the number of vertices to be anonymized and the security parameter k, and investigate the size of the graph (characterized by $|V|$) and the minimum cost r required to achieve k-security. The result shows that the cost r increases as the input graph becomes larger. Figure 4.6 supports the above observation from another perspective: when r and the number of vertices to be secured are fixed, the smaller a graph is, the easier it is to achieve a high level of security.

Similarly, the set A of vertices to be secured affects the cost of security positively. Figure 4.5 and Figure 4.8 together show the intuitive result that for the same graph, achieving a higher-level of security becomes more difficult when more vertices are added to A. Related to the this observation, Figure 4.9 demonstrates that to achieve the same level of security with the same cost, as the graph becomes larger, the set A needs to be smaller. We can see that at first the size of A is equal to the size of V because the fixed cost r is sufficient to reach k-security for all vertices in V. However as the size of graph grows, k-security can only be reached with cost r for a smaller set A.

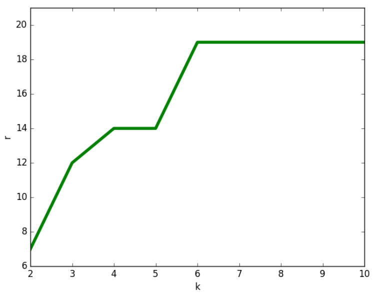

Figure 4.4: Relationship between k and r for network graphs

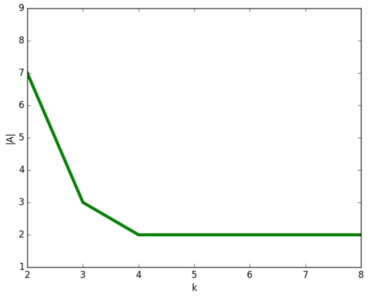

Figure 4.5: Relationship between k and $|A|$ for network graphs

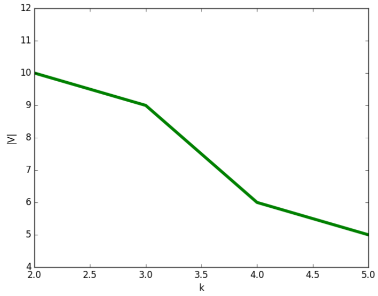

Figure 4.6: Relationship between k and $|V|$ for network graphs

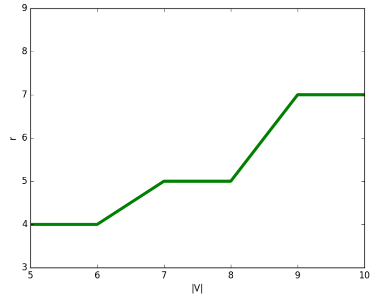

Figure 4.7: Relationship between $|V|$ and r for network graphs

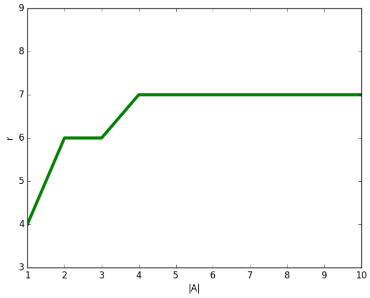

Figure 4.8: Relationship between $|A|$ and r for network graphs

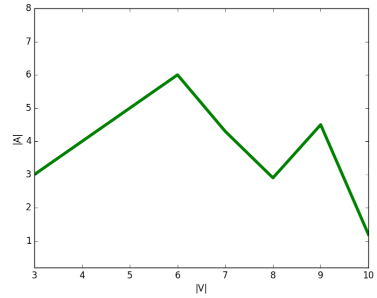

Figure 4.9: Relationship between $|V|$ and $|A|$ for network graphs

Chapter 5

Related Work

Many security notions have been proposed for graphs under different attack models in recent year. Most of these notions were defined in the context of social network, as it is a prominent use case of graphs. Their results however invariably apply to graphs in other contexts. The notions are often inspired by *k-anonymity* [21] defined for traditional database table. The actual definitions and techniques however differ in the context of graphs.

Liu and Terzi [15] for example proposed *k-degree-anonymity* in order to to prevent degree attack. They assumed that the attacker has prior knowledge of the degrees of some vertices in the graph. Addition and deletion of edges are used in this case to achieve security. Hay et al. [8] considered *vertex refinement attack*, which is a generalization of the degree attack, and proposed an anonymization approach utilizing random sampling, which retains integrity of summary data of the graph. Thompson and Yao [23] also define security against degree attack, but in their study the attacker is assumed to know about the degrees of all neighbours within distance i of some vertices. Zheleva and Getoor [26], on the other hand, considered security for social graphs whose edges are labelled but nodes are not. They used edge anonymization to prevent sensitive relationships from being revealed.

As Narayanan and Shmatikov [17] demonstrated, it is possible to conduct re-identification attack to a graph with only topological information. Therefore more recent notions consider attacks on structural information.

Zhou and Pei [27] defined neighbourhood attack in which the adversary tries to re-identify vertices in a graph with prior knowledge of their complete neighbourhood structure, and proposed a notion of *k-anonymity* by means of isomorphic neighbourhoods. The technique used to achieve security was inserting edges.

A more recent study by Wu et al. [24] proposed *k-symmetry*, which promises security against all structural attacks. The security notion proposed by [24] is very similar to that in [28], which was termed *k-automorphism*. These two notions are both based on the graph isomorphism problem, and the similarity was acknowledged by the authors. However the relationship between these two notions is unclear.

Imeson et al. [10] studies graph security in the context of integrated circuits. Their security notion is based on the problem of subgraph isomorphism, in this work we discuss a more general problem than they have studied.

Other works such as Blocki et al. [4] study graph security from the perspective of differential privacy. These works however focus on providing summary statistics of the graph instead of the graph itself. As already mentioned in Chapter 1 the isomorphism-based security notions we discuss in this work do not guarantee differential privacy.

Dedicated studies were conducted on the subject of RDF security. Rachapalli et al. [19] for example proposed operations for sanitizing vertices, edges and paths in an RDF graph. Their work was motivated by [3], which discusses RDF sanitization and anonymization from a practical standpoint. However, no clear notion of security was provided in these works. There has been other researches [12] on RDF access control, but they invariably focus on operations instead of security notions and are less relevant to this work.

Another important aspect of this study is the correctness of information. We base this part of our work on the research of RDF semantics conducted by ter Horst [22] and Arenas et al. [1], which discuss RDF interpreation and entailment in detail.

Chapter 6

Conclusion

Although graph anonymization have been studied in the context of mathematical graphs, the results do not necessarily apply to RDF graphs, which is a data model of directed, labeled graphs used in practical graph databases.

In this work we make two important contributions to bridge the gap between theory and practice for RDF graph security. The first part of our research compare established security notions in the traditional context. The four security notions we investigated are: *k-subgraph-isomorphism*, *k-neighbourhood-isomorphism*, *k-automorphism*, and *k-symmetry*. What these security notions have in common is that they are all based on graph isomorphism or related problems, and provide security against structural attacks to graphs. We propose a intuitive approach to compare the strength of these security notions, which is logical implication. We also provide complexity results for the problems of deciding whether a graph satisfies these security notions, which serve as lower bounds for the hardness of achieving these security notions. We establish that the strength of a security notion is uncorrelated with the hardness of enforcing the security notion. This part of the study serves as the basis for defining security for RDF graphs, but it also applies to graph security in contexts other than RDF.

As a result of the comparative study, we choose *k-automorphism* and *k-symmetry* to be adapted for RDF. Our adaptation fully takes into account the nature of RDF and caters to practical requirements. Our most important contribution in this context is that we consider not only security, but also the quality of information after sanitization. We ensure the correctness of information by requiring that the anonymization process follows the entailment relationship defined in RDF semantics. Moreover, we quantify the cost of security on information by the distance between the original graph and the anonymized graph on the lattice induced by entailment. As achieving *k-automorphism* or *k-symmetry* is computationally hard, we provide a Cnf-Sat reduction of these problems and conduct an empirical research with the reduction. Our empirical study sheds light on the relationship between information and security. It also shows that the structure of a graph can affect the cost of security.

References

[1] Marcelo Arenas, Mariano Consens, and Alejandro Mallea. Revisiting blank nodes in rdf to avoid the semantic mismatch with sparql. In *RDF Next Steps Workshop*, pages 26–27, 2010.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] Matt Bishop, Justin Cummins, Sean Peisert, Anhad Singh, Bhume Bhumiratana, Deborah Agarwal, Deborah Frincke, and Michael Hogarth. Relationships and data sanitization: a study in scarlet. In *Proceedings of the 2010 workshop on New security paradigms*, pages 151–164. ACM, 2010.

[4] Jeremiah Blocki, Avrim Blum, Anupam Datta, and Or Sheffet. Differentially private data analysis of social networks via restricted sensitivity. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 87–96. ACM, 2013.

[5] LLC Civic Impulse. Govtrack. https://www.govtrack.us/. Accessed: 2016-07-01.

[6] Richard Cyganiak, David Wood, and Markus Lanthaler. Rdf 1.1 concepts and abstract syntax. *W3C Recommendation. Feb*, 2014.

[7] Cynthia Dwork. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.

[8] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Philipp Weis. Resisting structural re-identification in anonymized social networks. *Proceedings of the VLDB Endowment*, 1(1):102–114, 2008.

[9] Patrick Hayes and Brian McBride. Rdf semantics, 2014.

[10] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V Tripunitara. Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation. In *USENIX Security*, volume 13, 2013.

[11] Complexible Inc. Stardog: Enterprise data unification with smart graphs. http://stardog.com/. Accessed: 2016-07-01.

[12] Amit Jain and Csilla Farkas. Secure resource description framework: an access control model. In *Proceedings of the eleventh ACM symposium on Access control models and technologies*, pages 121–129. ACM, 2006.

[13] Apache Jena. Apache jena. *jena. apache. org [Online]. Available: http://jena. apache. org [Accessed: Mar. 20, 2014]*, 2013.

[14] Zhiyuan Lin and Mahesh Tripunitara. From isomorphism-based security for graphs to semantics-preserving security for the resource description framework (rdf). *IEEE Transactions on Dependable and Secure Computing*, under review.

[15] Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 93–106. ACM, 2008.

[16] Anna Lubiw. Some np-complete problems similar to graph isomorphism. *SIAM Journal on Computing*, 10(1):11–21, 1981.

[17] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 173–187. IEEE, 2009.

[18] Eric Prudï£¡ï£¡Hommeaux, Andy Seaborne, et al. Sparql query language for rdf. *W3C recommendation*, 15, 2008.

[19] Jyothsna Rachapalli, Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham. Towards fine grained rdf access control. In *Proceedings of the 19th ACM symposium on Access control models and technologies*, pages 165–176. ACM, 2014.

[20] Pavan Reddivari, Tim Finin, and Anupam Joshi. Policy-based access control for an rdf store. In *Proceedings of the Policy Management for the Web workshop*, volume 120, pages 78–83, 2005.

[21] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[22] Herman J ter Horst. Completeness, decidability and complexity of entailment for rdf schema and a semantic extension involving the owl vocabulary. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):79–115, 2005.

[23] Brian Thompson and Danfeng Yao. The union-split algorithm and cluster-based anonymization of social networks. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, pages 218–227. ACM, 2009.

[24] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, and Zhihui Wang. K-symmetry model for identity anonymization in social networks. In *Proceedings of the 13th international conference on extending database technology*, pages 111–122. ACM, 2010.

[25] Viktor N Zemlyachenko, Nickolay M Korneenko, and Regina I Tyshkevich. Graph isomorphism problem. *Journal of Soviet Mathematics*, 29(4):1426–1481, 1985.

[26] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, security, and trust in KDD*, pages 153–171. Springer, 2008.

[27] Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 506–515. IEEE, 2008.

[28] Lei Zou, Lei Chen, and M Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *Proceedings of the VLDB Endowment*, 2(1):946–957, 2009.